Text Classification with Python

Learning Outcomes

By the end of this topic, you will have achieved the following learning outcomes:

- I can understand the concept of text classification.
- I can understand the importance of text classification.
- I can perform text classification to group data into different categories.

"If you want to understand people, especially your customers, then you have to be able to possess a strong capability to analyze text." Paul Hofmann

Reading

What is Text Classification?

Text classification is the process of assigning categories to text, based on its content. It's one of the fundamental tasks in Natural Language Processing with applications in sentiment analysis, topic labelling, spam detection and intent detection.

A few good examples are; articles can be organized by topics, support tickets can be organized by urgency, chat conversations can be organized by language, brand mentions can be organized by sentiment, and so on.

Why is Text Classification Important?

Some of the reasons why companies are leveraging text classification (with machine learning) are the following:

- **Scalability**
    - Manually analyzing and organizing text takes time. It's a slow process where a human needs to read each text and decide how to structure it.

Automated text classification changes this and enables us to easily analyze millions of texts at a fraction of a cost.

- **Real-time analysis**
    - There are critical situations that companies need to identify as soon as possible and take immediate action (e.g. PR crises on social media). Text classifiers with machine learning can make accurate precisions in real-time that enable companies to identify critical information instantly and take action right away.
- **Consistent criteria**
    - Human annotators make mistakes when classifying text data due to distractions, fatigue, and boredom. Other errors are generated due to inconsistent criteria. In contrast, automated text classification applies the same lens and criteria to all of the data, thus allowing humans to reduce errors with centralized text classification models.

How does Text Classification Work?

In order to understand how text classification works, we need to understand the different categories of text classification systems:

1. **Rule-based Systems**

   These types of systems classify text into organized groups by using a set of handcrafted linguistic rules. These rules instruct the system to use semantically relevant elements of a text to identify relevant categories based on its content.

   Example

   Say that we want to classify news articles into 2 groups, namely, Sports and Politics. First, we'll need to define two lists of words that characterize each group (e.g. words related to sports such as football, basketball, Lionel Messi, etc., and words related to politics such as Barack Obama, Putin, etc.).

   Next, when we want to classify a new incoming text, we'll need to count the number of sport-related words that appear in the text and do the same for politics-related words. If the number of sport-related word appearances is greater than the number of politics-related word count, then the text is classified as sports and vice versa.

   For example, this rule-based system will classify the headline "When is Lionel Messi' first game with Barcelona?" as Sports because it counted 1 sport-related term (Lionel Messi) and it didn't count any politics-related terms.

Rule-based systems are human comprehensible and can be improved over time. But this approach has some disadvantages.

- They are also time-consuming, since generating rules for a complex system can be quite challenging and usually requires a lot of analysis and testing.
- Rule-based systems are also difficult to maintain and don't scale well given that adding new rules can affect the results of the pre-existing rules.
- These systems also require deep knowledge of the domain.

2. **Machine Learning-Based Systems**
   These types of systems make classifications based on past observations. By **using pre-labelled examples** as training data, a machine learning algorithm can learn the different associations between pieces of text and that a particular output (i.e. tags) is expected for a particular input (i.e. text).

   The crucial step before training a classifier with machine learning is feature extraction: a method is used to transform each text into a numerical representation in the form of a vector. One of the most frequently used approaches is a bag of words, where a vector represents the frequency of a word in a predefined dictionary of words.

   > *For example, if we have defined our dictionary to have the following words {This, is, the, not, awesome, bad, basketball}, and we wanted to vectorize the text, we would have the following vector representation of that text: (1, 1, 0, 0, 1, 0, 0). Then, the machine learning algorithm is fed with training data that consists of pairs of feature sets (vectors for each text example) and tags (e.g. sports, politics) to produce a classification model.*

   If we want to make a topic classification model more accurate, we can just categorize more data, and retrain our model without having to go through complex procedures.

Steps of Performing Topic Classification

Step 1: Loading our data
Step 2: Data Exploration
Step 3: Data Preparation

- We perform basic data cleaning techniques and later perform text processing techniques. Below are some common text processing techniques.
  - **Lettercasing:** Converting all letters to either uppercase or lowercase.

    - **Spelling correction:** We can perform spelling correction to ensure that we are able to capture the words with the same meaning across the text we're analyzing.
    - **Splitting concatenated words:** Just as in spelling correction, when we split concatenated words such as "theywent" to "they" and "went", we are able to get words which are useful and would otherwise be lost if the step was not taken.
    - **Tokenizing:** Turning the texts into tokens. Tokens are words separated by spaces in a text.
    - **Stopword removal:** Some words do not contribute much to the machine learning model, so it's good to remove them. These words are called stop words. A list of stopwords can be defined by the NLTK library, or it can be business-specific and language specific.
    - **Normalization:** Normalization generally refers to a series of related tasks meant to put all text on the same level. Converting text to lowercase, removing special characters, and removing stop words will remove basic inconsistencies. Normalization improves text matching.
    - **Stemming:** Eliminating affixes (circumfixes, suffixes, prefixes, infixes) from a word in order to obtain a word stem. Porter Stemmer is the most widely used technique because it is very fast. Generally, stemming chops off the end of the word, and mostly it works fine.
      Example: Working -> Work
    - **Lemmatization:** The goal is the same as with stemming, but stemming a word sometimes loses the actual meaning of the word. Lemmatization usually refers to doing things properly using vocabulary and morphological analysis of words. It returns the base or dictionary form of a word, also known as the lemma.
      Example: Better -> Good.
    - **Feature extraction:** Algorithms use mathematics to train machine learning models. We now prepare our data by converting our text to numbers. There are several approaches i.e. Bag of Words and *TF-IDF.*
      NB: If we have multiple features, amongst them text features and other numerical features, we simply perform feature extraction to those text features and as a result get sparse matrices which we can include as numerical features in data modeling.
- During the process of data preparation, we can also generate meta features from our text, a process that allows us to have more features in our data. Below are such meta features.
    - ***Text Based Features***

        i. Number of words in the text
        ii. Number of unique words in the text
        iii. Number of characters in the text
        iv. Number of stopwords
        v. Number of punctuations
        vi. Number of uppercase words
        vii. Number of title case words
        viii. Average length of the words
    - ***NLP Based Features***
        i. Polarity
        ii. Subjectivity
        iii. Noun Count
        iv. Verb Count
        v. Adjective Count
        vi. Adverb Count
        vii. Pronoun Count
- Other types of features that we could also create include:
    - **Word Embedding Features**
        - These are features generated as a result of the use of learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers. We will cover these in depth in a future session.
    - **Topic Modelling Features**
        - Here, we would find labels upon performing clustering and generate a new feature. This will also be covered in a future session.

Step 4: Data Modeling

If we have many features with different scales, we would be required to perform scaling, later split our dataset and build a classifier using various classification algorithms. Below are some commonly used classification algorithms:

- **Naive Bayes**
    - **Naive Bayes (NB)** is a family of classification algorithms – the most popular for topic classification being Multinomial Naive Bayes (MNB) – that delivers great results when dealing with small amounts of data, say between 1,000 and 10,000 texts that work by correlating the probability of words appearing in a text with the probability of that text being about a certain topic.
- **Support Vector Machines (SVM)**

    - The advantage of SVM is that they often deliver better results than Naive Bayes for topic classification; the downside is that they require complex programming and require more computing resources.
    - However, it's possible to speed up the training process of an SVM by optimizing the algorithm by feature selection, in addition to running an optimized linear kernel such as sci-kit-learn's Linear SVC.
- **Logistic Regression Classifier, Random Forest, K Nearest Neighbors, Gradient Boosting, etc.**

Metrics and Evaluation
In this step, we test the actual category for a specific text and compare it to the predicted category then, with the results, calculate the following evaluation metrics:

- **Accuracy:** the percentage of texts that were assigned the correct topic.
- **Precision:** the percentage of texts the classifier classified correctly out of the total number of texts it predicted for each topic
- **Recall** is the percentage of texts the model predicted for each topic out of the total number of texts it should have predicted for that topic.
- **F1 Score:** the average of both precision and recall.

Making Predictions
Just like with any supervised learning problem, then use our created base model to make predictions.

Step 5: Model Optimisation
We can perform hyperparameter tuning, feature engineering techniques or even feed more data to our train set in order to improve the performance of our models.

Step 6: Model Deployment
This step would involve putting our model to a production environment where we would allow users to be able to use it.

Tools

- **NLTK** is a python library that is widely used for text classification. It provides plenty of resources to use for training models, plus different tools for processing text, including tokenization, stemming, tagging, parsing, and semantic reasoning.

- **SpaCy** is the fastest framework for training NLP models. Although it is less flexible and supports fewer languages than NLTK, it's much easier to use. SpaCy also provides built-in word vector.
- **Scikit-learn** provides a wide variety of algorithms for building machine learning models. It has excellent documentation and intuitive methods that make it easy to train a model for topic modelling.

References

You can also use the following resources for further reading.

1. Text Classification [Link]
2. Feature Extraction while Performing Text Classification [Link]
3. Introduction to Topic Modeling [Link]